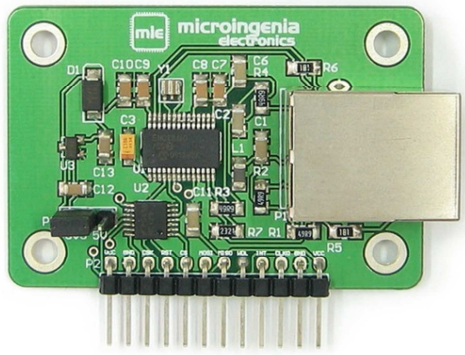
# Módulo de Comunicación Ethernet ENC28J60

## 1. Descripción del producto

Este módulo de desarrollo Ethernet, formado por el controlador ENC28J60 de Microchip, es una solución perfecta para crear una interfaz de red con la que controlar tu aplicación de forma remota a través de una red IP, ya sea mediante una página web o a través del envío de correos electrónicos para advertir de incidencias.

Integra la electrónica necesaria para ser conectado directamente a un Router/Switch mediante un cable de red. El control se realiza mediante el bus serie SPI.

El módulo puede ser alimentado tanto a 5V como a 3,3V, para ello dispone de un jumper de selección de la tensión de alimentación. El conector Ethernet presenta además dos leds (verde y naranja) para indicar la correcta conexión al router y el estado de la transmisión de datos.



## 2. Características generales

- Controlador Ethernet ENC28J60
- IEEE 802.3
- 10BASE-T
- Full/Half Duplex

## 3. Especificaciones

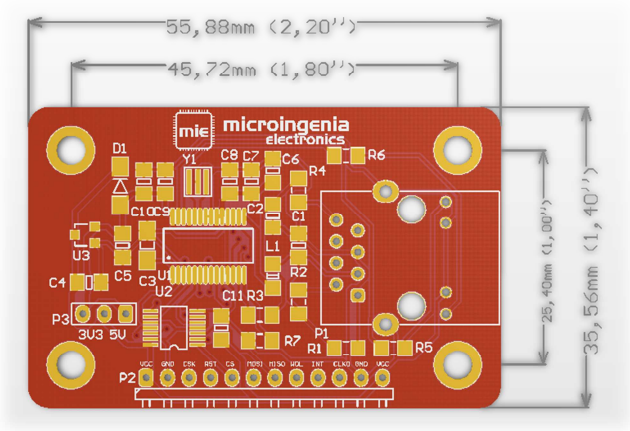
- Alimentación: 3.3V o 5 VDC
- Consumo: 138mA@3.3V, 145mA@5V
- Dimensiones: 55,88 x 35,56mm (2.2'' x 1.4'')
- Peso: 16,6g

## 4. Ideas de aplicación

- Aplicaciones de Comunicación
- Envío de parámetros por e-mail
- Visualización de parámetros en una sencilla web

**NOTA:** El esquemático, y los códigos de ejemplo se pueden descargar desde la web del producto (ModEthernet) a través de www.microingenia.com

## 5. Descripción de Pines




| Pin | Descripción |
|---|---|
| VCC | Conexión de alimentación 5V o 3V3 |
| GND | Conexión de masa |
| CSK | Entrada de reloj SPI interface |
| RST | Pin de Reset a nivel bajo |
| CS | Selección del chip mediante SPI |
| MOSI | Pin de entrada de datos SPI |
| MISO | Pin de salida de datos SPI |
| WOL | No conectado |
| INT | Pin de interrupción |
| CLKO | Reloj de salida programable |

**Figura 1.- Dimensiones ModEthernet**

## 6. Esquemático

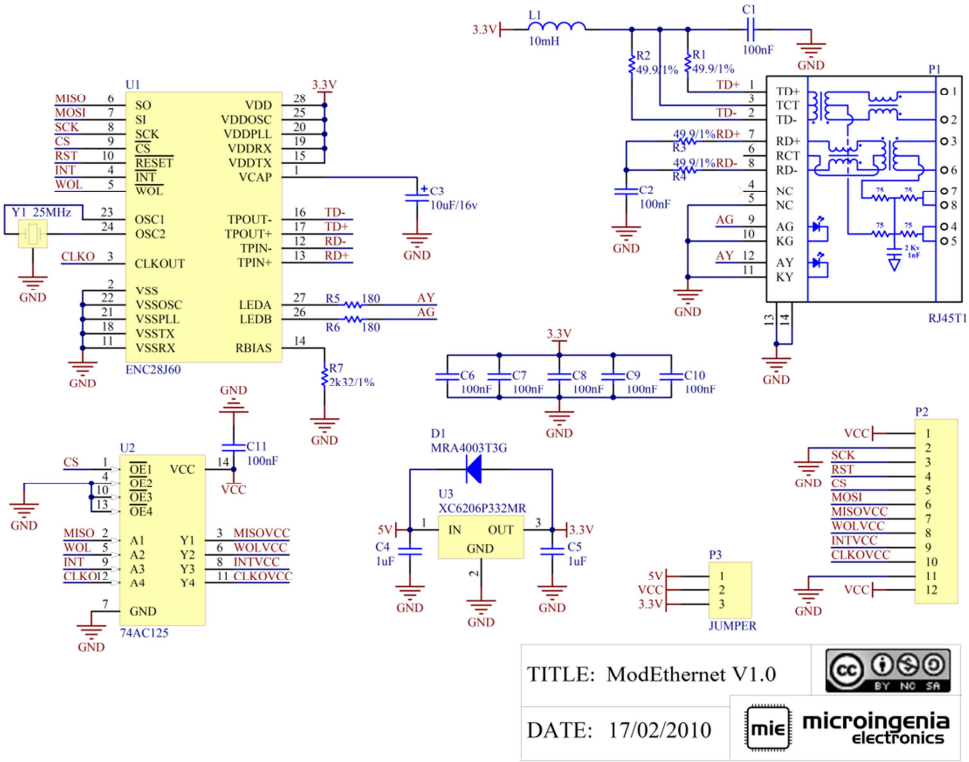



**Figura 2.- Esquemático ModEthernet**

- **Componentes principales:**
  - **U1:** ENC28J60: Módulo de Comunicación Ethernet ENC28J60
  - **U2:** 74AC125: Buffer tri-estado de cuatro salidas
  - **U3:** Regulador 5V->3.3V
  - **RJ45T1:** Conector Ethernet
  - **Jumper:** Selección de tipo de alimentación (5V o 3.3V)

## 7. Ejemplo 1:

- **Objetivo**

Ejemplo de diseño para el envío de un correo electrónico mediante el uso del módulo y la entrenadora 18F67J50Trainer, definiendo en el programa las direcciones de envío, destino y los parámetros necesarios para el envío del correo.

- **Conexión Módulo-Entrenadora**

La conexión entre la entrenadora y el modulo se realizará siguiendo el esquema de conexiones definido en el fichero “config18F67J50Trainer.c”.

```
// Connections in 18F67J50Trainer
#define  LedV      PIN_G0
#define  LedR      PIN_G1
#define  NewMail   PIN_B0

// Connections between ModEthernet and 18F67J50Trainer
#define  PIN_ENC_MAC_CS     PIN_D0
#define  PIN_ENC_MAC_RST    PIN_D1
#define  PIN_ENC_MAC_INT    PIN_D2
#define  PIN_ENC_MAC_WOL    PIN_D3
#define  PIN_ENC_MAC_SI     PIN_D4   // MISO: PIC > ENC
#define  PIN_ENC_MAC_SO     PIN_D5   // MOSI: PIC < ENC
#define  PIN_ENC_MAC_CLK    PIN_D6   // SCK

// Bootloader
// PIN_C0 - GND -> Btld Mode
// PIN_C0 - VCC -> Code Mode
```

**Figura 3- Conexión de pines ModEthernet**

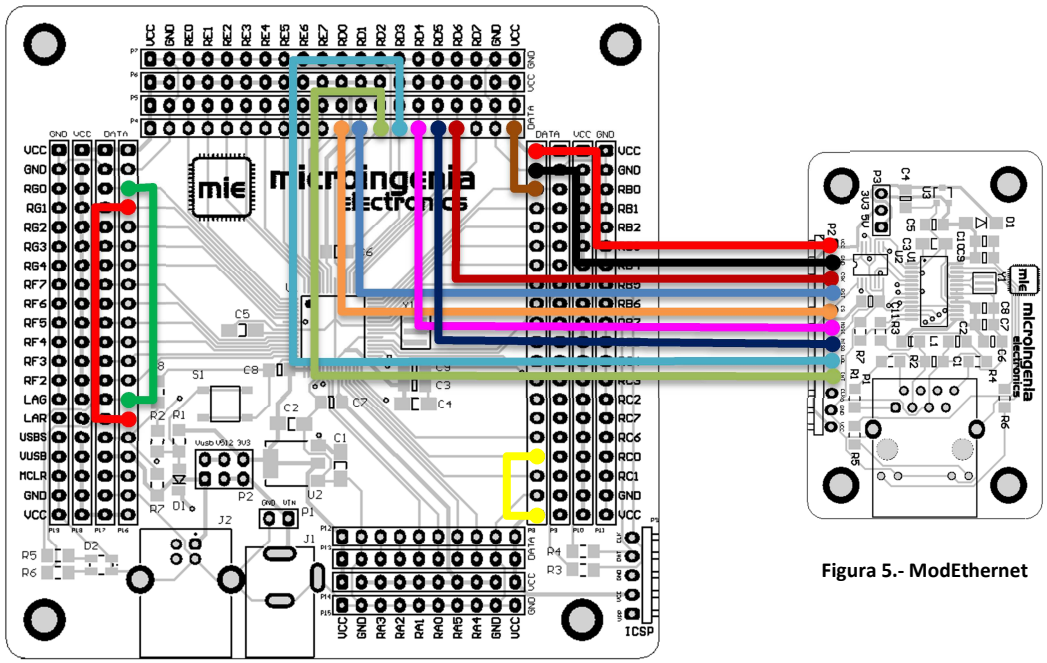


**Figura 4.- 18F67J50Trainer**



**Figura 5.- ModEthernet**

- **Código de Aplicación en C para el compilador de CCS PCWHD Versión 4.108**

El código de ejemplo CCS disponible en la web (Ejemplo1 V1.0, CCS C "E1_ModEthernet_CCS_V1.0") permite el envío de correo electrónico mediante el uso del módulo de una forma sencilla variando unos sencillos parámetros, de esta forma el envío se realiza de forma prácticamente inmediata, en el caso de que se produzca algún tipo de error el código devuelve donde se ha producido.

A lo largo del código se explican de forma más detallada las funciones que se han utilizado para el envío.

```
#include "config18F67J50Trainer.c"

#define  STACK_USE_MCPENC      TRUE
#define  STACK_USE_ARP         TRUE
#define  STACK_USE_DHCP        TRUE
#define  STACK_USE_DNS         TRUE
#define  STACK_USE_ICMP        TRUE
#define  __ESMTP
#define  STACK_USE_BASE64      TRUE
#define  STACK_USE_TCP         TRUE
#define  STACK_USE_UDP         TRUE

#include "tcpip/stacktsk.c"
#include "tcpip/smtp.c"

#include "usb/usb_bootloader.h"
#include "usb/usb_cdc.h"

// Editor de tus datos de e-mail, los datos son orientativos, se deben incluir datos
reales
#define SMTP_HOST_NAME        "smtp.microingenia.com"
#define SMTP_PORT             "port"
#define SMTP_USERNAME         "test@microingenia.com"
#define SMTP_PWD              "test"
#define EMAIL_FROM_ADDRESS    "test@microingenia.com"
#define EMAIL_TO_ADDRESS      "test@microingenia.com"
#define EMAIL_SUBJECT         "eMail TEST"
#define EMAIL_BODY            "Hi PIC World!"

int8 state = 0;
int8 timeoutEthernet = 0;
int1 tick = FALSE;
int1 flag_sendinMailEth = FALSE;
int1 flag_emailSent = FALSE;
int1 flag_sendmail = FALSE;

IP_ADDR smtpserver;
int16 port;
char hostname[64];
char username[64];
char pwd[64];
char from[64];
char to[64];
char subject[64];
char body[64];
```

```
void MACAddrInit(void)
{
   MY_MAC_BYTE1=0x00;
   MY_MAC_BYTE2=0x04;
   MY_MAC_BYTE3=0xA3;
   MY_MAC_BYTE4=0x00;
   MY_MAC_BYTE5=0x00;
   MY_MAC_BYTE6=0x01;
}

// Configuración del e-mail con los parámetros indicados en la definición anterior
void SendMailConfig(void)
{
   // Fijamos el Puerto SMTP
   port = 25;

   // Nombre del Host del servidor SMTP
   sprintf(hostname, SMTP_HOST_NAME);

   // Nombre de usuario y contraseña del servidor SMTP
   sprintf(username, SMTP_USERNAME);
   sprintf(pwd, SMTP_PWD);

   //Dirección de envío y destino del correo electrónico
   sprintf(from, EMAIL_FROM_ADDRESS);
   sprintf(to, EMAIL_TO_ADDRESS);

   //Asunto del correo
   sprintf(subject, EMAIL_SUBJECT);

   //Cuerpo del correo
   sprintf(body, EMAIL_BODY);
}

// Función encargada de seguir los pasos para el envío del correo electrónico, en caso
de error indicará en que parámetro se ha producido

void tarea_ethernet(void)
{
   StackTask();
   if(flag_sendinMailEth) SMTPTask();

   switch(state)
   {
      case 0:
         if(!MACIsLinked())
         {
            if(!tick)
            {
               printf(usb_cdc_putc,"\n\rNo Ethernet Link");
               tick = TRUE;
            }
         }
         else
         {
            state++;
            tick = FALSE;
         }
         break;
      case 1:
         if(!DHCPIsBound())
```

```
         {
            if(!tick)
            {
               printf(usb_cdc_putc,"\n\rDCHP Not Bound");
               tick = TRUE;
            }
         }
         else
         {
            printf(usb_cdc_putc,"\n\rIP:    %u.%u.%u.%u",    MY_IP_BYTE1,    MY_IP_BYTE2,
MY_IP_BYTE3, MY_IP_BYTE4);
            printf(usb_cdc_putc,"\n\rMASK:  %u.%u.%u.%u",  MY_MASK_BYTE1,  MY_MASK_BYTE2,
MY_MASK_BYTE3, MY_MASK_BYTE4);
            printf(usb_cdc_putc,"\n\rGATE:  %u.%u.%u.%u",  MY_GATE_BYTE1,  MY_GATE_BYTE2,
MY_GATE_BYTE3, MY_GATE_BYTE4);
            printf(usb_cdc_putc,"\n\rDNS:   %u.%u.%u.%u",   MY_DNS_BYTE1,   MY_DNS_BYTE2,
MY_DNS_BYTE3, MY_DNS_BYTE4);
            state++;
            tick = FALSE;
         }
         break;
      case 2:
         if(!tick)
         {
            DNSResolve(hostname);
            printf(usb_cdc_putc,"\n\rFinding IP address for %s", SMTP_HOST_NAME);
            tick = TRUE;
         }
         else
         {
            state++;
            tick = FALSE;
         }
         timeoutEthernet = 0;
         break;
      case 3:
         if(DNSIsResolved(&smtpserver))
         {
            printf(usb_cdc_putc,"\n\rIP   address   is   resolved   to   ""%u.%u.%u.%u""",
smtpserver.v[0], smtpserver.v[1], smtpserver.v[2], smtpserver.v[3]);
            timeoutEthernet = 0;
            state++;
         }
         else if(timeoutEthernet > 1)
         {
            printf(usb_cdc_putc,"\n\rDNS failure %X.", smDNS);
            timeoutEthernet = 0;
            state = 0;
         }
         break;
      case 4:
         if(SMTPConnectAndStart(&smtpserver, port, username, pwd, from, to, subject))
         {
            printf(usb_cdc_putc,"\n\rOpening socket with SMTP server...");
            flag_sendinMailEth=1;
            state++;
         }
         else
         {
            printf(usb_cdc_putc,"\n\rSMTP   engine   is   busy!    Try   again   in   a   few
moments.");
```

```
            state = 0;
         }
         break;
      case 5:
         if (SMTPLastError())
         {
            printf(usb_cdc_putc,"\n\rSMTP Error %X", SMTPLastError());
            state = 0;
         }
         else if (SMTPIsPutReady())
         {
            printf(usb_cdc_putc,"\n\rSMTP  Connection  accepted,  now  sending  body  of
email.");
            printf(SMTPPut, "%s", body);
            SMTPDisconnect();
            state++;
         }
         break;
      case 6:
         if (SMTPIsFree())
         {
            if (!SMTPLastError())
            {
               flag_emailSent = TRUE;
               flag_sendmail = FALSE;
               printf(usb_cdc_putc,"\n\rE-mail Sent!");
            }
            else
               printf(usb_cdc_putc,"\n\rE-Mail    rejected    by    SMTP    server!    %X",
SMTPLastError());
            state = 0;
         }
         break;
   }
}

//Inicio de los parámetros del módulo
void ethernet_init(void)
{
   MACAddrInit();
   StackInit();
   SendMailConfig();
   SMTPInit();
}

// Función principal del programa, iniciando el Puerto USB, el programa espera recibir
la confirmación de un nuevo e-mail.
//Se iniciará el módulo de red con los parámetros introducidos en la configuración
inicial, y se invocará la función tarea_ethernet() de forma recursiva hasta el término
de la misma.
//Cuando el correo ha sido enviado correctamente el programa devolverá “e-mail sent”, si
se ha producido un error en la función, nos devolverá mediante un código de error el
parámetro de la función donde se ha producido dicho fallo.

void main(void)
{
   PLLEN = 1;

   ON(LedR);
   OFF(LedV);
```

```
   usb_cdc_init();
   usb_init();
   usb_wait_for_enumeration();

   ON(LedV);
   OFF(LedR);

   while(!input(NewMail));
   printf(usb_cdc_putc,"\n\rSendMail V1.0 Example:");

   ethernet_init();
   flag_sendmail = TRUE;

   while(TRUE)
   {
      if(flag_sendmail) tarea_ethernet();
      else
         if(input(NewMail)) flag_sendmail = TRUE;
   }
}
```

## 8. Ejemplo 2

- **Objetivo**

El código de ejemplo CCS disponible en la web (Ejemplo1 V1.0, CCS C "E2_ModEthernet_CCS_V1.0") permite la creación de una sencilla página web en la memoria interna del PIC, accesible desde la dirección 192.168.1.111

- **Conexión Módulo-Entrenadora**

La conexión entre la entrenadora y el modulo se realizará siguiendo el esquema de conexiones definido en el fichero "config18F4550Trainer.c".

```
// Connections in 18F4550Trainer
#define  LedV  PIN_B0
#define  LedR  PIN_B1

// Connections between ModEthernet and 18F4550Trainer
#define  PIN_ENC_MAC_SO    PIN_D5   // MOSI: PIC < ENC
#define  PIN_ENC_MAC_SI    PIN_D4   // MISO: PIC > ENC
#define  PIN_ENC_MAC_CLK   PIN_D6   // SCK
#define  PIN_ENC_MAC_CS    PIN_D0
#define  PIN_ENC_MAC_RST   PIN_D1
#define  PIN_ENC_MAC_INT   PIN_D2
#define  PIN_ENC_MAC_WOL   PIN_D3

// Bootloader
// PIN_C0 - GND -> Btld Mode
// PIN_C0 - VCC -> Code Mode
```

**Figura 6- Conexión de pines ModEthernet**

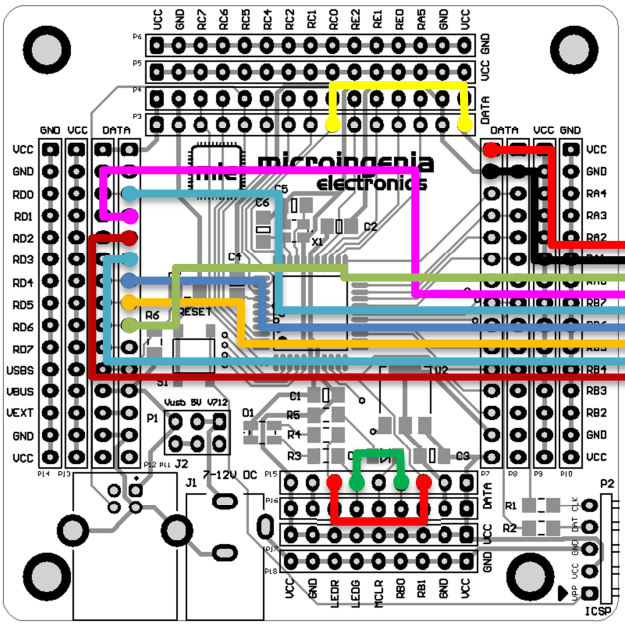


Figura 7.- 18F4550Trainer

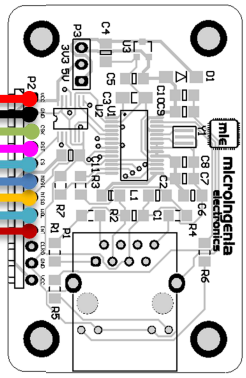


Figura 8.- ModEthernet

- **Código de Aplicación en C para el compilador de CCS PCWHD Versión 4.106**

```
#include "config18F4550Trainer.c"

#define  STACK_USE_MCPENC  TRUE
#define  STACK_USE_ARP     TRUE
#define  STACK_USE_ICMP    TRUE
#define  STACK_USE_TCP     TRUE

#include "tcpip/stacktsk.c"

#include "usb/usb_bootloader.h"
#include "usb/usb_cdc.h"

#define MY_HTTP_SOCKET  80

int8 HTTPSocket=INVALID_SOCKET;

// Microchip VendorID
// MAC: 00-04-A3-XX-XX-XX
void MACAddrInit(void)
{
   MY_MAC_BYTE1=0x00;
   MY_MAC_BYTE2=0x04;
   MY_MAC_BYTE3=0xA3;
   MY_MAC_BYTE4=0x00;
   MY_MAC_BYTE5=0x00;
   MY_MAC_BYTE6=0x01;
}

void IPAddrInit(void)
{
   //Dirección IP
```

```
   MY_IP_BYTE1=192;
   MY_IP_BYTE2=168;
   MY_IP_BYTE3=1;
   MY_IP_BYTE4=111;

   //Pasarela de red
   MY_GATE_BYTE1=192;
   MY_GATE_BYTE2=168;
   MY_GATE_BYTE3=1;
   MY_GATE_BYTE4=1;

   //Mascara de subred
   MY_MASK_BYTE1=255;
   MY_MASK_BYTE2=255;
   MY_MASK_BYTE3=255;
   MY_MASK_BYTE4=0;
}

void HTTPPut(char c)
{
   TCPPut(HTTPSocket, c);
}

// Funcion para crear el cuerpo de la web, establece la conexión inicial con la página,
la creación de la página se realiza mediante un sencillo código HTML

void HTTPTask(void)
{
   static enum {HTTP_ST_CONNECT=0, HTTP_ST_WAIT_CONNECT=1, HTTP_ST_GET=2, HTTP_ST_PUT=3,
HTTP_ST_DISCONNECT=4} state=0;
   static TICKTYPE timeout_counter;
   static char lc, lc2;
   char c;
   char content[250];

   if (HTTPSocket==INVALID_SOCKET)
      state=HTTP_ST_CONNECT;
   else if (!TCPIsConnected(HTTPSocket))
      state=HTTP_ST_WAIT_CONNECT;
   else if (TickGetDiff(TickGet(), timeout_counter) > TICKS_PER_SECOND*60)
      state=HTTP_ST_DISCONNECT;

   switch(state) {
      case HTTP_ST_CONNECT:
         HTTPSocket=TCPListen(MY_HTTP_SOCKET);
         if (HTTPSocket!=INVALID_SOCKET) {
            //printf("\r\nHTTP: Listening");
            state=HTTP_ST_WAIT_CONNECT;
            timeout_counter=TickGet();
         }
         break;

      case HTTP_ST_WAIT_CONNECT:
         timeout_counter=TickGet();
         if (TCPIsConnected(HTTPSocket)) {
            state=HTTP_ST_GET;
            //printf("\r\nHTTP: Connected");
         }
         break;

      case HTTP_ST_GET:
```

```
         if (TCPIsGetReady(HTTPSocket)) {
            while (TCPGet(HTTPSocket, &c)) {
               if ( (c=='\n') && (lc2=='\n') ) {
                  state=HTTP_ST_PUT;
                  //printf("\r\nHTTP: Got Page Request");
               }
               lc2=lc;
               lc=c;
            }
         }
         break;

      case HTTP_ST_PUT:
         if (TCPIsPutReady(HTTPSocket)) {
            //printf("\r\nHTTP: Putting Page Response");
            sprintf(content,                              "<HTML><HEAD><TITLE>Microingenia
Electronics</TITLE></HEAD><BODY><H1>Test ModEthernet V1.0</H1>");
            sprintf(&content[0]+strlen(content),"</BODY></HTML>");
            printf(HTTPPut, "HTTP/1.1 200 OK\r\n");
            printf(HTTPPut, "Content-Type: text/html\r\n");
            printf(HTTPPut, "Content-Length: %u\r\n",strlen(content));
            printf(HTTPPut, "\r\n");
            printf(HTTPPut, "%s", content);
            //printf("\r\nHTTP: Flushing...");
            TCPFlush(HTTPSocket);
            //printf("\r\nHTTP: Flushed!");
            state=HTTP_ST_GET;
         }
         break;

      case HTTP_ST_DISCONNECT:
         if (TCPIsPutReady(HTTPSocket)) {
            //printf("\r\nHTTP: Force Disconnect");
            TCPDisconnect(HTTPSocket);
            state=HTTP_ST_WAIT_CONNECT;
         }
         break;
   }
}

void main(void)
{
   ON(LedR);
   OFF(LedV);

   MACAddrInit();
   IPAddrInit();
   StackInit();

   ON(LedV);
   OFF(LedR);

   while(TRUE)
   {
      StackTask();
      HTTPTask();
   }
}
```